Modelling the Compatibility of Licenses

Benjamin Moreau, Patricia Serrano-Alvarado, Matthieu Perrin, Emmanuel Desmontils

Modelling the Compatibility of Licenses

Benjamin Moreau[1,2], Patricia Serrano-Alvarado[1], Matthieu Perrin[1], and Emmanuel Desmontils[1]

[1] Nantes University, LS2N, CNRS, UMR6004, 44000 Nantes, France
{Name.LastName@}univ-nantes.fr
[2] OpenDataSoft {Name.Lastname}@opendatasoft.com

Abstract. Web applications facilitate combining resources (linked data, web services, source code, documents, etc.) to create new ones. For a resource producer, choosing the appropriate license for a combined resource is not easy. It involves choosing a license compliant with all the licenses of combined resources and analysing the reusability of the resulting resource through the compatibility of its license. The risk is either, to choose a license too restrictive making the resource difficult to reuse, or to choose a not enough restrictive license that will not sufficiently protect the resource. Finding the right trade-off between compliance and compatibility is a difficult process. An automatic ordering over licenses would facilitate this task. Our research question is: *given a license l_i, how to automatically position l_i over a set of licenses in terms of compatibility and compliance?* We propose CaLi, a model that partially orders licenses. Our approach uses restrictiveness relations among licenses to define compatibility and compliance. We validate experimentally CaLi with a quadratic algorithm and show its usability through a prototype of a license-based search engine. Our work is a step towards facilitating and encouraging the publication and reuse of licensed resources in the Web of Data.

1 Introduction

Web applications facilitate combining resources (linked data, web services, source code, documents, etc.) to create new ones. To facilitate reuse, resource producers should systematically associate licenses with resources before sharing or publishing them [1]. Licenses specify precisely the conditions of reuse of resources, i.e., what actions are *permitted*, *obliged* and *prohibited* when using the resource.

For a resource producer, choosing the appropriate license for a combined resource or choosing the appropriate licensed resources for a combination is a difficult process. It involves choosing a license compliant with all the licenses of combined resources as well as analysing the reusability of the resulting resource through the compatibility of its license. The risk is either, to choose a license too restrictive making the resource difficult to reuse, or to choose a not enough restrictive license that will not sufficiently protect the resource.

Relations of compatibility, compliance and restrictiveness on licenses could be very useful in a wide range of applications. Imagine license-based search en-

gines for services such as GitHub[3], APISearch[4], LODAtlas[5], DataHub[6], Google Dataset Search[7] or OpenDataSoft[8] that could find resources licensed under licenses compatible or compliant with a specific license. Answers could be partially ordered from the least to the most restrictive license. We argue that a model for license orderings would allow the development of such applications.

We consider simplified definitions of compliance and compatibility inspired by works like [2–5]: *a license l_j is compliant with a license l_i if a resource licensed under l_i can be licensed under l_j without violating l_i*. If a license l_j is compliant with l_i then we consider that l_i is compatible with l_j and that resources licensed under l_i are reusable with resources licensed under l_j. In general, if l_i is compatible with l_j then l_j is more (or equally) restrictive than l_i. We also consider that *a license l_j is more (or equally) restrictive than a license l_i if l_j allows at most the same permissions and has at least the same prohibitions/obligations than l_i*.

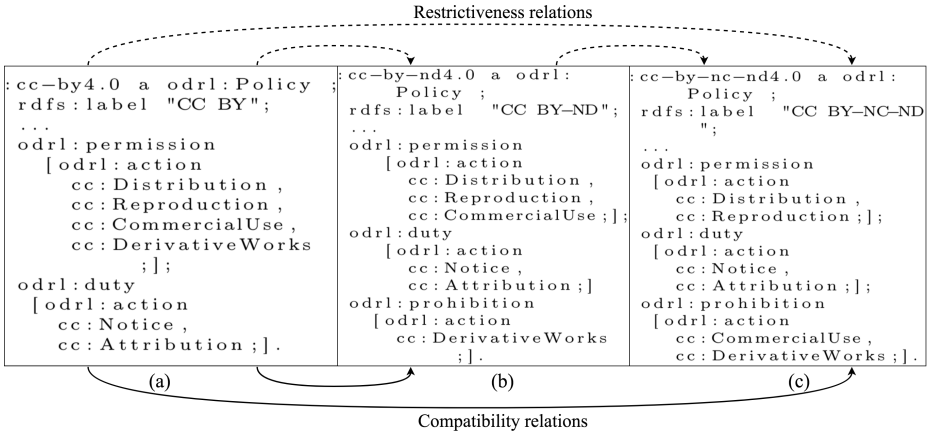

Fig. 1: Three Creative Commons licenses described in RDF.

Usually but not always, when l_i is less restrictive than l_j then l_i is compatible with l_j. For instance, see Fig. 1 that shows an excerpt of three Creative Commons (CC)[9] licenses described in RDF and using the ODRL vocabulary[10]. Notice that there exists a restrictiveness order among these licenses, (a) is less restrictive than (b) and (b) is less restrictive than (c). By transitivity (a) is less restrictive than (c). Notice also that (a) is compatible with (b) and (c), but (b)

[3] `https://github.com/`
[4] `http://apis.io/`
[5] `http://lodatlas.lri.fr/`
[6] `https://datahub.io/`
[7] `https://toolbox.google.com/datasetsearch`
[8] `https://data.opendatasoft.com/`
[9] `https://creativecommons.org/`
[10] The term *duty* is used for obligations `https://www.w3.org/TR/odrl-model/`

is not compatible with (c). This is due to the semantics of the prohibited action *DerivativeWorks* that forbids the distribution of a derivation (remix, transform or build upon) of the protected resource under a different license. Thus, depending on the semantics of their actions, a restrictiveness relation between two licenses does not imply a compatibility relation.

Our research question is: *given a license l_i, how to automatically position l_i over a set of licenses in terms of compatibility and compliance?* The challenge we face is how to generalise the automatic definition of the ordering relations among licenses while taking into account the influence of the semantics of actions.

Inspired by lattice-based access control models [6,7], we propose CaLi (ClAssification of LIcenses), a model for license orderings that uses restrictiveness relations and constraints among licenses to define compatibility and compliance. We validate experimentally CaLi with a quadratic algorithm and show its usability through a prototype of a license-based search engine. Our work is a step towards facilitating and encouraging the publication and reuse of licensed resources in the Web of Data. However, it is not intended to provide legal advice.

This paper is organised as follows. Section 2 discuses related works, Section 3 introduces the CaLi model, Section 4 illustrates the usability of our model, Section 5 shows experiments of the implemented algorithm as well as the prototype of a license-based search engine, and Section 6 concludes.

2 Related work

Automatic license classification requires machine-readable licenses. License expression languages such as CC REL[11], ODRL, or L4LOD[12] enable fine-grained RDF description of licenses. Works like [8] and [9] use natural language processing to automatically generate RDF licenses from licenses described in natural language. Other works such as [10–12] propose a set of well-known licenses in RDF described in CC REL and ODRL. Thereby, in this work, we suppose that there exist consistent licenses described in RDF.

There exist some tools to facilitate the creation of license compliant resources. TLDRLegal[13], CC Choose[14] and ChooseALicense[15] help users to choose actions to form a license for their resources. CC search[16] allows users to find images licensed under Creative Commons licenses that can be commercialized, modified, adapted, or built upon. Web2rights proposes a tool to check compatibility among Creative Commons licenses[17]. DALICC [12] allows to compose arbitrary licenses and provides information about equivalence, similarity and compatibility of licenses. Finally, Licentia[18], based on deontic logic to reason over the licenses,

[11] `https://creativecommons.org/ns`
[12] `https://ns.inria.fr/l4lod/`
[13] `https://tldrlegal.com/`
[14] `https://creativecommons.org/choose/`
[15] `https://choosealicense.com/`
[16] `https://ccsearch.creativecommons.org/`
[17] `http://www.web2rights.com/creativecommons/`
[18] `http://licentia.inria.fr/`

proposes a web service to find licenses compatible with a set of permissions, obligations and prohibitions chosen by the user. From these tools, only Licentia and DALICC use machine-readable licenses[19,20] in RDF. But unfortunately, these works do not order licenses in terms of compatibility or compliance.

The easiest way to choose a license for a combined resource is to create a new one by combining all resource licenses to combine. Several works address the problem of license compatibility and license combination. In web services, [2] proposes a framework that analyses compatibility of licenses to verify if two services are compatible and then generates the composite service license. [13] addresses the problem of license preservation during the combination of digital resources (music, data, picture, etc.) in a collaborative environment. Licenses of combined resources are combined into a new one. In the Web of Data, [3] proposes a framework to check compatibility among CC REL licenses. If licenses are compatible, a new license compliant with combined ones is generated. [4] formally defines the combination of licenses using deontic logic. [14] proposes PrODUCE, an approach to combine usage policies taking into account the usage context. These works focus on combining operators for automatic license combination but do not propose to position a license over a set of licenses.

Concerning the problem of license classification to facilitate the selection of a license, [15] uses Formal Concept Analysis (FCA) to generate a lattice of actions. Once pruned and annotated, this lattice can be used to classify licenses in terms of features. This classification reduces the selection of a license to an average of three to five questions. However, this work does not address the problem of license compatibility. Moreover, FCA is not suitable to generate compatibility or restrictiveness relations among licenses. FCA defines a derivation operator on objects that returns a set of attributes shared by the objects. We consider that the set of actions in common of two licenses is not enough to infer these relations. If applied to our introductory example, FCA can only work with permissions but not with obligations and prohibitions. That is because l_i is less restrictive than l_j if permissions of l_i are a superset of permissions of l_j, but regarding obligations and prohibitions, l_i is less restrictive than l_j if they are a subset of those of l_j. In the context of Free Open Source Software (FOSS), [5] proposes an approach, based on a directed acyclic graph, to detect license violations in existing software packages. It considers that license l_i is compatible with l_j if the graph contains a path from l_i to l_j. However, as such a graph is build from a manual interpretation of each license, its generalisation and automation is not possible.

In the domain of access control, [6] proposes a lattice model of secure information flow. This model classifies security classes with associated resources. Like in the compatibility graph of [5], security class sc_i is compatible with sc_j if the lattice contains a path from sc_i to sc_j. Thus, this path represents the authorized flow of resources (e.g., resource r_i protected with sc_i can flow to a resource protected by sc_j without violating sc_i.). The lattice can be generated automatically through a pairwise combination of all security classes if sc_i combined with sc_k

[19] `http://rdflicense.appspot.com/rdflicense`
[20] `https://www.dalicc.net/license-library`

gives sc_j where sc_i and sc_k are both compatible with sc_j. [7] describes several models based on this approach but none focuses on classifying licenses.

None of these works answers our research question. They do not allow to automatically position a license over a set of licenses in terms of compatibility or compliance. In our work we propose a lattice-based model inspired by [6]. This model is independent of any license description language, application context and licensed resource so that it can be used in a wide variety of domains.

3 CaLi: a lattice-based license model

The approach we propose to partially order licenses in terms of compatibility and compliance passes through a restrictiveness relation. In a license, actions can be distributed in what we call *status*, e.g., permissions, obligations and prohibitions. To decide if a license l_i is less restrictive than l_j, it is necessary to know if an action in a status is considered as less restrictive than the same action in another status. In the introductory example (Fig. 1), we consider that permissions are less restrictive than obligations, which are less restrictive than prohibitions, i.e., $Permission \leqslant Duty \leqslant Prohibition$. This relation can be seen in Fig 2b.

We remark that if two licenses have a restrictiveness relation then it is possible that they have a compatibility relation too. The restrictiveness relation between the licenses can be automatically obtained according to the status of actions without taking into account the semantics of the actions. Thus, based on lattice-ordered sets [16], we define a restrictiveness relation among licenses.

To identify the compatibility among licenses, we refine the restrictiveness relation with constraints. The goal is to take into account the semantics of actions. Constraints also distinguish valid licenses from non-valid ones. We consider a license l_i as non-valid if a resource can not be licensed under l_i, e.g., a license that simultaneously permits the *Derive* action[21] and prohibits *DerivativeWorks*[22].

This approach is based on:

1. a set of *actions* (e.g., *read, modify, distribute*, etc.);
2. a *restrictiveness lattice of status* that defines (i) all possible status of an action in a license (i.e., permission, obligation, prohibition, recommendation, undefined, etc.) and (ii) the restrictiveness relation among status; a *restrictiveness lattice of licenses* is obtained from a combination of 1 and 2;
3. a set of *compatibility constraints* to identify if a restrictiveness relation between two licenses is also a compatibility relation; and
4. a set of *license constraints* to identify non-valid licenses.

Next section introduces formally the CaLi model and Section 3.2 introduces a simple example of a CaLi ordering.

[21] `https://www.w3.org/TR/odrl-vocab/#term-derive`
[22] `https://www.w3.org/TR/odrl-vocab/#term-DerivativeWorks`

3.1 Formal model description

We first define a *restrictiveness lattice of status*. We use a lattice structure because it is necessary, for every pair of status, to know which status is less (or more) restrictive than both.

Definition 1 (Restrictiveness lattice of status $\mathcal{LS}$).
A restrictiveness lattice of status *is a lattice $\mathcal{LS} = (S, \leqslant_S)$ that defines all possible* status *S for a license and the relation $\leqslant_S$ as the* restrictiveness relation *over S. For two status s_i, s_j, if $s_i \leqslant_S s_j$ then s_i is less restrictive than s_j.*

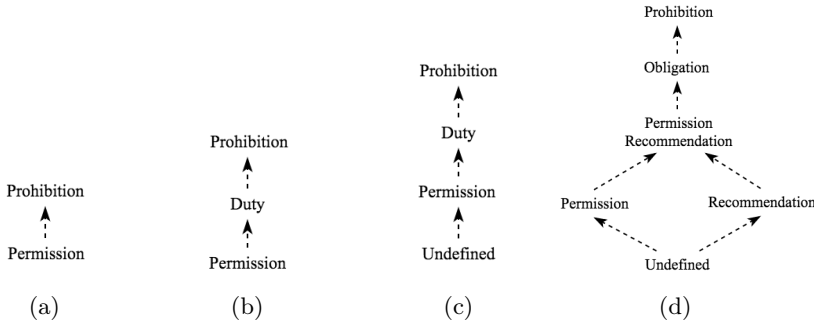

Fig. 2: Examples of restrictiveness lattices of status ($\mathcal{LS}$). Dashed arrows represent restrictiveness, e.g., in (a) Permission is less restrictive than Prohibition.

Different $\mathcal{LS}$s can be defined according to the application domain. Fig. 2a shows the diagram of a $\mathcal{LS}$ inspired by file systems where actions can be either prohibited or permitted. With this lattice, prohibiting to read a file is more restrictive than permitting to read it. Fig. 2b illustrates a $\mathcal{LS}$ for CC licenses where actions are either permitted, required (Duty) or prohibited. Fig. 2c shows a $\mathcal{LS}$ inspired by the ODRL vocabulary. In ODRL, actions can be either permitted, obliged, prohibited or not specified (i.e., undefined). In this lattice, the undefined status is the least restrictive and the prohibited one the most restrictive. Fig. 2d shows a $\mathcal{LS}$ where a recommended or permitted action is less restrictive than the same action when it is permitted and recommended.

Now we formally define a license based on the status of its actions.

Definition 2 (License).
Let $\mathcal{A}$ be a set of actions and $\mathcal{LS} = (S, \leqslant_S)$ be a restrictiveness lattice of status. A license *is a function $l : \mathcal{A} \to S$. We denote by $\mathcal{L}_{\mathcal{A},\mathcal{LS}}$ the set of all licenses.*

For example, consider $\mathcal{A} = \{read, modify, distribute\}$, $\mathcal{LS}$ the lattice of Fig. 2c and two licenses: l_i which permits *read* and *distribute* but where *modify* is undefined and l_j where *modify* is also undefined but which permits *read* and prohibits *distribute*. We define l_i and l_j as follows:

$\forall a \in \mathcal{A}$:

$$l_i(a) = \begin{cases} \text{Undefined} & \text{if } a \in \{\textit{modify}\}; \\ \text{Permission} & \text{if } a \in \{\textit{read}, \textit{distribute}\}. \end{cases}$$

$$l_j(a) = \begin{cases} \text{Undefined} & \text{if } a \in \{\textit{modify}\}; \\ \text{Permission} & \text{if } a \in \{\textit{read}\}; \\ \text{Prohibition} & \text{if } a \in \{\textit{distribute}\}. \end{cases}$$

A restrictiveness lattice of status and a set of licenses make possible to partially order licenses in a restrictiveness lattice of licenses.

Definition 3 (Restrictiveness relation over licenses).
Let A be a set of actions and $\mathcal{LS} = (S, \leqslant_S)$ be a restrictiveness lattice of status associated to the join and meet operators $\vee_S$ and $\wedge_S$, and $l_i, l_j \in \mathcal{L}_{\mathcal{A},\mathcal{LS}}$ be two licenses. We say that l_i is less restrictive than l_j, denoted $l_i \leqslant_{\mathcal{R}} l_j$, if for all actions $a \in \mathcal{A}$, the status of a in l_i is less restrictive than the status of a in l_j. That is, $l_i \leqslant_{\mathcal{R}} l_j$ if $\forall a \in \mathcal{A}, l_i(a) \leqslant_S l_j(a)$.

Moreover, we define the two operators $\vee$ and $\wedge$ as follows. For all actions $a \in \mathcal{A}$, the status of a in $l_i \vee l_j$ (resp. $l_i \wedge l_j$) is the join (resp. meet) of the status of a in l_i and the status of a in l_j. That is, $(l_i \vee l_j)(a) = l_i(a) \vee_S l_j(a)$ and $(l_i \wedge l_j)(a) = l_i(a) \wedge_S l_j(a)$.

For example, consider $\mathcal{LS}$ the lattice of Fig. 2c, and licenses l_i and l_j defined previously; $l_i \leqslant_{\mathcal{R}} l_j$ because $l_i(\textit{read}) \leqslant_S l_j(\textit{read}), l_i(\textit{modify}) \leqslant_S l_j(\textit{modify})$ and $l_i(\textit{distribute}) \leqslant_S l_j(\textit{distribute})$. In this example, $l_i \vee l_j = l_j$ because $\forall a \in \mathcal{A}$, $(l_i \vee l_j)(a) = l_j(a)$, e.g., $(l_i \vee l_j)(\textit{distribute}) = l_j(\textit{distribute}) = \textit{Prohibition}$. If for an action, it is not possible to say which license is the most restrictive then the compared licenses are not comparable by the restrictiveness relation.

Remark 1 *The pair $(\mathcal{L}_{\mathcal{A},\mathcal{LS}}, \leqslant_{\mathcal{R}})$ is a restrictiveness lattice of licenses, whose $\vee$ and $\wedge$ are respectively the join and meet operators.*

In other words, for two licenses l_i and l_j, $l_i \vee l_j$ (resp. $l_i \wedge l_j$) is the least (resp. most) restrictive license that is more (resp. less) restrictive than both l_i and l_j.

Remark 2 *For an action $a \in \mathcal{A}$, we call $(\mathcal{L}_{\{a\},\mathcal{LS}}, \leqslant_{\mathcal{R}})$ the* action lattice *of a. Remark that $(\mathcal{L}_{\mathcal{A},\mathcal{LS}}, \leqslant_{\mathcal{R}})$ and $\prod_{a\in\mathcal{A}}(\mathcal{L}_{\{a\},\mathcal{LS}}, \leqslant_{\mathcal{R}})$ are isomorphic. That is, a restrictiveness lattice of licenses can be generated through the coordinatewise product [16] of all its* action lattices. *The total number of licenses in this lattice is $|\mathcal{LS}|^{|\mathcal{A}|}$.*

For example, consider $\mathcal{A} = \{\textit{read}, \textit{modify}\}$, $\mathcal{LS}$ the lattice of Fig. 2a, $(\mathcal{L}_{\mathcal{A},\mathcal{LS}}, \leqslant_{\mathcal{R}})$ is isomorphic to $(\mathcal{L}_{\{\textit{read}\},\mathcal{LS}}, \leqslant_{\mathcal{R}}) \times (\mathcal{L}_{\{\textit{modify}\},\mathcal{LS}}, \leqslant_{\mathcal{R}})$. Figure 3a,b,c illustrates the product of these action lattices and the produced restrictiveness lattice of licenses.

To identify the compatibility relation among licenses and to distinguish valid licenses from non-valid ones it is necessary to take into account the semantics of actions. Thus, we apply two types of constraints to the restrictiveness lattice of licenses: license constraints and compatibility constraints.

Definition 4 (License constraint).
Let $\mathcal{L}_{\mathcal{A},\mathcal{LS}}$ be a set of licenses. A license constraint is a function $\omega_{\mathcal{L}} : \mathcal{L}_{\mathcal{A},\mathcal{LS}} \rightarrow Boolean$ which identifies if a license is valid or not.

For example, the license constraint $\omega_{\mathcal{L}_1}$ considers a license $l_i \in \mathcal{L}_{\mathcal{A},\mathcal{LS}}$ non-valid if *read* is prohibited but modification is permitted (i.e., a *modify* action implies a *read* action):

$$\omega_{\mathcal{L}_1}(l_i) = \begin{cases} False \text{ if } l_i(read) = \text{Prohibition and } l_i(modify) = \text{Permission;} \\ True \text{ otherwise.} \end{cases}$$

Definition 5 (Compatibility constraint).
Let $(\mathcal{L}_{\mathcal{A},\mathcal{LS}}, \leqslant_{\mathcal{R}})$ be a restrictiveness lattice of licenses. A compatibility constraint is a function $\omega_{\rightarrow} : \mathcal{L}_{\mathcal{A},\mathcal{LS}} \times \mathcal{L}_{\mathcal{A},\mathcal{LS}} \rightarrow Boolean$ which constraints the restrictiveness relation $\leqslant_{\mathcal{R}}$ to identify compatibility relations among licenses.

For example, consider that a license prohibits the action *modify*. In the spirit of *DerivativeWork*, we consider that the distribution of the modified resource under a different license is prohibited. Thus, the compatibility constraint $\omega_{\rightarrow_1}$, considers that a restrictiveness relation $l_i \leqslant_{\mathcal{R}} l_j$ can be also a compatibility relation if l_i does not prohibit *modify*. This constraint is described as:

For $l_i, l_j \in \mathcal{L}_{\mathcal{A},\mathcal{LS}}$,

$$\omega_{\rightarrow_1}(l_i, l_j) = \begin{cases} False \text{ if } l_i(modify) = \text{Prohibition;} \\ True \text{ otherwise.} \end{cases}$$

Now we are able to define a CaLi ordering from a restrictiveness lattice of licenses and constraints defined before.

Definition 6 (CaLi ordering).
A CaLi ordering is a tuple $\langle \mathcal{A}, \mathcal{LS}, C_{\mathcal{L}}, C_{\rightarrow} \rangle$ such that $\mathcal{A}$ and $\mathcal{LS}$ form a restrictiveness lattice of licenses $(\mathcal{L}_{\mathcal{A},\mathcal{LS}}, \leqslant_{\mathcal{R}})$, $C_{\mathcal{L}}$ is a set of license constraints and $C_{\rightarrow}$ is a set of compatibility constraints. For two licenses $l_i \leqslant_{\mathcal{R}} l_j \in \mathcal{L}_{\mathcal{A},\mathcal{LS}}$, we say that l_i is compatible with *l_j, denoted by $l_i \rightarrow l_j$, if $\forall \omega_{\mathcal{L}} \in C_{\mathcal{L}}, \omega_{\mathcal{L}}(l_i) = \omega_{\mathcal{L}}(l_j) = True$ and $\forall \omega_{\rightarrow} \in C_{\rightarrow}, \omega_{\rightarrow}(l_i, l_j) = True$.*

Remark 3 *We define the* compliance relation *as the opposite of the* compatibility relation. *For two licenses l_i, l_j, if $l_i \rightarrow l_j$ then l_j* is compliant with *l_i.*

A CaLi ordering is able to answer our research question, *given a license l_i, how to automatically position l_i over a set of licenses in terms of compatibility and compliance?* It allows to evaluate the potential reuse of a resource depending on its license. Knowing the compatibility of a license allows to know to which extent the protected resource is reusable. On the other hand, knowing the compliance of a license allows to know to which extent other licensed resources can be reused. Next section shows an example of CaLi ordering.

3.2 Example 1

Consider a CaLi ordering $\langle \mathcal{A}, \mathcal{LS}, \{\omega_{\mathcal{L}_1}\}, \{\omega_{\rightarrow_1}\}\rangle$ such that:

- $\mathcal{A}$ is the set of actions $\{read,\ modify\}$,
- $\mathcal{LS}$ is a restrictiveness lattice of status where an action can be either permitted or prohibited, and $Permission \leqslant_S Prohibition$ (cf Fig. 2a),
- $\omega_{\mathcal{L}_1}$ is the license constraint introduced in the example of Def. 4, and
- $\omega_{\rightarrow_1}$ is the compatibility constraint introduced in the example of Def. 5.

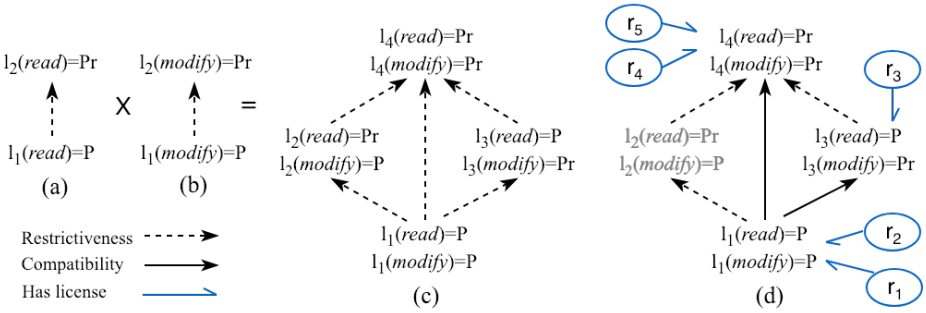

Fig. 3: (a) and (b) are the action latices $(\mathcal{L}_{\{read\},\mathcal{LS}}, \leqslant_{\mathcal{R}})$ and $(\mathcal{L}_{\{modify\},\mathcal{LS}}, \leqslant_{\mathcal{R}})$, where $\mathcal{A} = \{read, modify\}$ and $\mathcal{LS}$ is the lattice of Fig. 2a (Pr=Prohibition and P=Permission). The product of these action lattices gives the restrictiveness lattice of licenses (c) $(\mathcal{L}_{\mathcal{A},\mathcal{LS}}, \leqslant_{\mathcal{R}})$ (reflexive relations are not represented). (d) is the CaLi ordering $\langle \mathcal{A}, \mathcal{LS}, \{\omega_{\mathcal{L}_1}\}, \{\omega_{\rightarrow_1}\}\rangle$.

Fig. 3d shows a visual representation of this CaLi ordering. Licenses in grey are identified as non-valid by $\omega_{\mathcal{L}_1}$. They are part of the ordering but cannot protect resources. Dashed arrows represent restrictiveness relations $\leqslant_{\mathcal{R}}$. Black arrows represent restrictiveness relations that are also compatibility relations.

Consider a set of resources $R = \{r_1, r_2, r_3, r_4, r_5\}$. $\rightharpoonup$ is the *has license* relation such that $\{r_1, r_2\} \rightharpoonup l_1; r_3 \rightharpoonup l_3; \{r_4, r_5\} \rightharpoonup l_4$. Thanks to our CaLi ordering, next questions can be answered.

- *Which licensed resources can be reused in a resource that has as license l_3?* Those resource whose licenses are compatible with l_3: r_1 and r_2 that have license l_1 which precedes l_3, as well as r_3 that has the license l_3 itself.
- *Which licensed resources can reuse a resource that has as license l_1?* Those resource whose licenses are compliant with l_1: r_3, r_4 and r_5 that have licenses l_3 and l_4 which follow l_1, as well as r_1 and r_2 that have the license l_3 itself.

Resulting licenses can be returned ordered in a graph of compatibility.

We illustrated CaLi with a simple restrictiveness lattice of status, next section introduces a more realistic CaLi ordering inspired by licenses of Creative Commons.

4 A CaLi ordering for Creative Commons

Creative Commons proposes 7 licenses that are legally verified, free of charge, easy-to-understand and widely used when publishing resources on the Web. These licenses use 7 actions that can be permitted, required or prohibited. In this CaLi example, we search to model a complete compatibility ordering of all possible valid licenses using these 7 actions.

4.1 Description of a CC ordering based on CaLi

Consider CC_CaLi, a CaLi ordering $\langle \mathcal{A}, \mathcal{LS}, C_{\mathcal{L}}, C_{\rightarrow} \rangle$ such that:

- $\mathcal{A}$ is the set of actions *{cc:Distribution, cc:Reproduction, cc:DerivativeWorks, cc:CommercialUse, cc:Notice, cc:Attribution, cc:ShareAlike}*,
- $\mathcal{LS}$ is the restrictiveness lattice of status depicted in 2b[23], and
- $C_{\mathcal{L}}, C_{\rightarrow}$ are the sets of constraints defined next.

$C_{\mathcal{L}} = \{\omega_{\mathcal{L}_2}, \omega_{\mathcal{L}_3}\}$ allows to invalidate a license (1) when *cc:CommercialUse* is required and (2) when *cc:ShareAlike* is prohibited:

$$\omega_{\mathcal{L}_2}(l_i) = \begin{cases} False \text{ if } l_i(\textit{cc:CommercialUse}) = \text{Duty;} \\ True \text{ otherwise.} \end{cases}$$

$$\omega_{\mathcal{L}_3}(l_i) = \begin{cases} False \text{ if } l_i(\textit{cc:ShareAlike}) = \text{Prohibition;} \\ True \text{ otherwise.} \end{cases}$$

$C_{\rightarrow} = \{\omega_{\rightarrow_2}, \omega_{\rightarrow_3}\}$ allows to identify (1) when *cc:ShareAlike* is required and (2) when *cc:DerivativeWorks* is prohibited. That is because *cc:ShareAlike* requires that the distribution of derivative works be under the same license only, and *cc:DerivativeWorks*, when prohibited, does not allow the distribution of a derivative resource, regardless of the license.

$$\omega_{\rightarrow_2}(l_i, l_j) = \begin{cases} False \text{ if } l_i(\textit{cc:ShareAlike}) = \text{Duty;} \\ True \text{ otherwise.} \end{cases}$$

$$\omega_{\rightarrow_3}(l_i, l_j) = \begin{cases} False \text{ if } l_i(\textit{cc:DerivativeWorks}) = \text{Prohibition;} \\ True \text{ otherwise.} \end{cases}$$

Other constraints could be defined to be closer to the CC schema[24] but for the purposes of this compatibility ordering these constraints are enough.

4.2 Analysis of CC_CaLi

The size of the restrictiveness lattice of licenses is 3^7 but the number of valid licenses of CC_CaLi is 972 due to $C_{\mathcal{L}}$. That is, 5 actions in whatever status and 2 actions (*cc:CommercialUse* and *cc:ShareAlike*) in only 2 status: $3^5 * 2^2$.

[23] To simplify, we consider that a requirement is a duty.

[24] `https://creativecommons.org/ns`

The following CC_CaLi licenses are like the official CC licenses.

$$CCBY(a) = \begin{cases} \text{Permission if } a \in \{\textit{cc:Distribution, cc:Reproduction} \\ \qquad\qquad\qquad\qquad \textit{cc:DerivativeWorks, cc:CommercialUse} \\ \qquad\qquad\qquad\qquad \textit{cc:ShareAlike}\}; \\ \text{Duty} \qquad \text{if } a \in \{\textit{cc:Notice, cc:Attribution}\}. \end{cases}$$

$$CCBYNC(a) = \begin{cases} \text{Permission} \;\; \text{if } a \in \{\textit{cc:Distribution, cc:Reproduction} \\ \qquad\qquad\qquad\qquad\quad \textit{cc:DerivativeWorks, cc:ShareAlike}\}; \\ \text{Duty} \qquad \text{if } a \in \{\textit{cc:Notice, cc:Attribution}\}; \\ \text{Prohibition if } a \in \{\textit{cc:CommercialUse}\}. \end{cases}$$

The following CC_CaLi licenses are not part of the official CC licenses. License $CC\ l_1$ is like CC BY-NC but without the obligation to give credit to the copyright holder/author of the resource. $CC\ l_2$ is like CC BY but with the prohibition of making multiple copies of the resource. License $CC\ l_3$ allows only exact copies of the original resource to be distributed. $CC\ l_4$ is like $CC\ l_3$ with the prohibition of commercial use.

$$CC\ l_1(a) = \begin{cases} \text{Permission} \;\; \text{if } a \in \{\textit{cc:Distribution, cc:Reproduction,} \\ \qquad\qquad\qquad\qquad\quad \textit{cc:DerivativeWorks, cc:ShareAlike,} \\ \qquad\qquad\qquad\qquad\quad \textit{cc:Notice, cc:Attribution}\}; \\ \text{Prohibition if } a \in \{\textit{cc:CommercialUse}\}. \end{cases}$$

$$CC\ l_2(a) = \begin{cases} \text{Permission} \;\; \text{if } a \in \{\textit{cc:Distribution, cc:DerivativeWorks,} \\ \qquad\qquad\qquad\qquad\quad \textit{cc:CommercialUse, cc:ShareAlike}\}; \\ \text{Duty} \qquad \text{if } a \in \{\textit{cc:Notice, cc:Attribution}\}; \\ \text{Prohibition if } a \in \{\textit{cc:Reproduction}\}. \end{cases}$$

$$CC\ l_3(a) = \begin{cases} \text{Permission} \;\; \text{if } a \in \{\textit{cc:Distribution, cc:ShareAlike, cc:CommercialUse}\}; \\ \text{Duty} \qquad \text{if } a \in \{\textit{cc:Notice, cc:Attribution, cc:Reproduction}\}; \\ \text{Prohibition if } a \in \{\textit{cc:DerivativeWorks}\}. \end{cases}$$

$$CC\ l_4(a) = \begin{cases} \text{Permission} \;\; \text{if } a \in \{\textit{cc:Distribution, cc:ShareAlike}\}; \\ \text{Duty} \qquad \text{if } a \in \{\textit{cc:Notice, cc:Attribution, cc:Reproduction}\}; \\ \text{Prohibition if } a \in \{\textit{cc:DerivativeWorks, cc:CommercialUse}\}. \end{cases}$$

In CC_CaLi, the minimum is the license where all actions are permitted (i.e., CC Zero) and the maximum is the license where all actions are prohibited.

Fig. 4 shows two subgraphs of CC_CaLi with only the compatibility relations. Fig. 4a shows only the 7 official CC licenses and Fig. 4b includes also $CC\ l_1$ to $CC\ l_4$. These graphs can be generated using the CaLi implementation (cf Section 5). Thanks to $\omega_{\to_2}$, the restrictiveness relation between CC BY-SA and CC BY-NC-SA is not identified as a compatibility relation and thanks to $\omega_{\to_3}$, the restrictiveness relation between CC BY-ND and CC BY-NC-ND is not identified as a compatibility relation. We recall that a license that prohibits *cc:DerivativeWorks* is not compatible even with itself.

The compatibility relations of Fig. 4a are conform to the ones obtained from the Web2rights tool. This example shows the usability of CaLi with a real set of licenses.

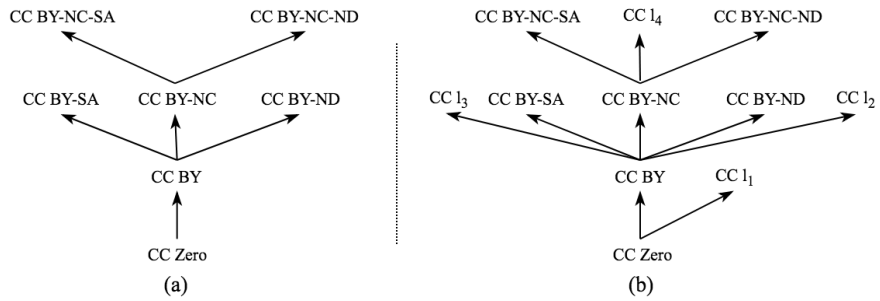

Fig. 4: Compatibility subgraphs of CC_CaLi: (a) contains the 7 official CC licenses and (b) contains $CC\ l_1$ to $CC\ l_4$ in addition to the 7 official CC licenses.

5 Implementation of CaLi orderings

The goal of this section is twofold, to analyse the algorithm we implemented to produce CaLi orderings and to illustrate the usability of CaLi through a prototype of a license-based search engine.

5.1 Experimental validation

The size growth of CaLi orderings is exponential, i.e., $|\mathcal{LS}|^{|\mathcal{A}|}$. Nevertheless, it is not necessary to explicitly build a CaLi ordering to use it. Sorting algorithms like insertion sort can be used to produce subgraphs of a CaLi ordering.

We implemented an algorithm that can sort any set of licenses using the $\mathcal{LS}$ of Fig. 2c in $\sum_{i=0}^{n-1} i = \frac{n(n-1)}{2}$ comparisons of restrictiveness (approx. $n^2/2$), n being the number of licenses to sort, i.e., $O(n^2)$. The goal is to be able to insert a license in a graph in linear time $O(n)$ without sorting again the graph.

We use a heuristic, based on the restrictiveness of the new license, to chose between two strategies, *1)* to insert a license traversing the graph from the minimum or *2)* from the maximum. To do this, our algorithm calculates the relative position of the new license (node) from the number of actions that it obliges and prohibits. The median depth (number of levels) of the existing graph is calculated from the median of the number of prohibited and obliged actions of existing licenses. Depending on these numbers, a strategy is chosen to find the place of the new license in the graph.

Results shown in Fig. 5 demonstrate that our algorithm sorts a set of licenses with at most $n^2/2$ comparisons. We used 20 subsets of licenses of different sizes from the CC_CaLi ordering. Size of subsets was incremented by 100 up to 2187 licenses. Each subset was created and sorted 3 times randomly. The curve was produced with the average of the number of comparisons to sort each subset.

A comparison of restrictiveness takes on average 6 milliseconds[25], thus to insert a license in a 2000 licenses graph takes an average of 12 seconds. Building

[25]With a 160xIntel(R) Xeon(R) CPU E7-8870 v4 2.10GHz 1,5 Tb RAM.

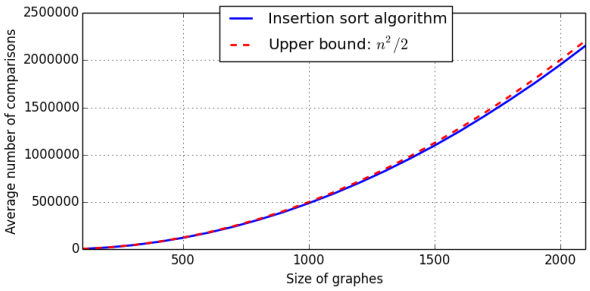

Fig. 5: Performance of the implemented insertion sort algorithm in number of comparisons of restrictiveness with incremental size of subsets of licenses.

a whole graph is time consuming (a 2000 licenses graph takes on average 8 hours to sort) but this time can be reduced with further optimisations of the process to compare the restrictiveness of two licenses. The implementation in Python of our algorithm and details of our experiments are available on GitHub[26].

5.2 A search engine based on an ODRL CaLi ordering

We implemented a prototype of a search engine that allows to find linked data[27] and source code repositories[28] based on the compatibility or the compliance of their licenses. We use licenses described with the ODRL vocabulary. ODRL proposes properties to define semantic dependencies among actions[29] that we translate as CaLi constraints. *Included In* is defined as "An Action transitively asserts that another Action encompasses its operational semantics". *Implies* is defined as "An Action asserts that another Action is not prohibited to enable its operational semantics". Thereby we consider that if an action a_i is included in another action a_j then a_i implies a_j. For example, *CommercialUse* is included in *use*, therefore we consider that *CommercialUse* implies *use*. That means that if *CommercialUse* is permitted then *use* should be permitted too. To preserve this dependency we implemented the constraint $\omega_{\mathcal{L}_4}$.

$$\omega_{\mathcal{L}_4}(l_i) = \begin{cases} False \text{ if } a_i \text{ odrl:includedIn } a_j \\ \qquad \text{AND } (l_i(a_i) = \text{Permitted OR } l_i(a_i) = \text{Obliged}) \\ \qquad \text{AND } l_i(a_j) = \text{Prohibited;} \\ True \quad \text{otherwise.} \end{cases}$$

We use $ODRL\_CaLi$, a CaLi ordering $\langle \mathcal{A}, \mathcal{LS}, C_{\mathcal{L}}, C_{\rightarrow} \rangle$ such that:

- $\mathcal{A}$ is the set of 72 actions of ODRL,
- $\mathcal{LS}$ is the restrictiveness lattice of status of Fig. 2c,
- $C_{\mathcal{L}} = \{\omega_{\mathcal{L}_2}, \omega_{\mathcal{L}_3}, \omega_{\mathcal{L}_4}\}$, and

[26] `https://github.com/benjimor/CaLi-Search-Engine`
[27] `http://cali.priloo.univ-nantes.fr/ld/`
[28] `http://cali.priloo.univ-nantes.fr/rep/`
[29] `https://www.w3.org/TR/odrl-vocab/#actionConcepts`

– $C_{\rightarrow} = \{\omega_{\rightarrow_2}, \omega_{\rightarrow_3}\}$.

The size of this ordering is 4^{72} and it is not possible to build it. This search engine illustrates the usability of $ODRL\_CaLi$ through two subgraphs. On the one side, there is a subgraph with the most used licenses in DataHub[30] and OpenDataSoft. Licenses in this graph are linked to some RDF datasets such that it is possible to find datasets whose licenses are compatible (or compliant) with a particular license. On the other side, there is a subgraph with the most used licenses in GitHub. Here, licenses are linked to some GitHub repositories and it is possible to find repositories whose licenses are compatible (or compliant) with a particular license.

Discussion The model we propose uses restrictiveness as the basis to define compatibility and compliance among licenses. This strategy works most of the time, as we have shown in this paper, but it has certain limitations. In particular, CaLi is not designed to define the compatibility of two licences if it is not coherent with their restrictiveness relation. As an example consider two versions of MPL licenses. Version 2.0 relaxes some obligations compared to version 1.1. Thus, MPL-2.0 is less restrictive than MPL-1.1. With CaLi constraints, it can only be possible to say that MPL-2.0 is compatible with MPL-1.1. But in the legal texts it is said the opposite, i.e., MPL-1.1 is compatible with MPL-2.0.

Thereby, particularities in the usage of compatibility of licenses, the granularity of the semantisation of licenses and the understanding of some actions (like *ShareAlike*) are the main reasons of the difference between CaLi orderings and other classifications. This is the case, for instance, of our compatibility graph devoted to licenses of GitHub and the graph presented in [5].

6 Conclusions and perspectives

We proposed a lattice-based model to define compatibility and compliance relations among licenses. Our approach is based on a restrictiveness relation that is refined with constraints to take into account the semantics of actions existing in licenses. We have shown the feasibility of our approach through two CaLi orderings, one using the Creative Commons vocabulary and the second using ODRL. We experimented the production of CaLi orderings with the implementation of an insertion sort algorithm whose cost is $n^2/2$. We implemented a prototype of a license-based search engine that highlights the feasibility and usefulness of our approach. Our compatibility model does not intent to provide a legal advice but it allows to exclude those licenses that would contravene a particular license.

A perspective of this work is to take into account other aspects of licenses related to usage contexts like jurisdiction, dates of reuse, etc. Another perspective is to analyse how two compatibility orderings can be compared. That is, given two CaLi orderings, if there is an alignment between their vocabularies and their restrictiveness lattices of status are homomorphic then find a function to pass from a CaLi ordering to another.

[30] `https://old.datahub.io/`

Acknowledgments Authors thank Margo Bernelin and Sonia Desmoulin-Canselier (laboratory of *Droit et Changement Social* - UMR CNRS 6297) for our helpful discussions on this work.

References

1. O. Seneviratne, L. Kagal, and T. Berners-Lee, "Policy-Aware Content Reuse on the Web," in *International Semantic Web Conference (ISWC)*, 2009.
2. G. Gangadharan, M. Weiss, V. D'Andrea, and R. Iannella, "Service License Composition and Compatibility Analysis," in *International Conference on Service-Oriented Computing (ICSOC)*, 2007.
3. S. Villata and F. Gandon, "Licenses Compatibility and Composition in the Web of Data," in *Workshop Consuming Linked Data (COLD) collocated with ISWC*, 2012.
4. G. Governatori, A. Rotolo, S. Villata, and F. Gandon, "One License to Compose Them All. A Deontic Logic Approach to Data Licensing on the Web of Data," in *International Semantic Web Conference (ISWC)*, 2013.
5. G. M. Kapitsaki, F. Kramer, and N. D. Tselikas, "Automating the License Compatibility Process in Open Source Software With SPDX," *Journal of Systems and Software*, vol. 131, 2017.
6. D. E. Denning, "A Lattice Model of Secure Information Flow," *Communications of the ACM*, vol. 19, no. 5, 1976.
7. R. S. Sandhu, "Lattice-Based Access Control Models," *Computer*, vol. 26, no. 11, 1993.
8. N. Sadeh, A. Acquisti, T. D. Breaux, L. F. Cranor, and et.al., "Towards Usable Privacy Policies: Semi-Automatically Extracting Data Practices from Websites' Privacy Policies," in *Symposium on Usable Privacy and Security (SOUPS)*, 2014. Poster.
9. E. Cabrio, A. P. Aprosio, and S. Villata, "These Are Your Rights," in *European Semantic Web Conference (ESWC)*, 2014.
10. V. Rodríguez Doncel, A. Gómez-Pérez, and S. Villata, "A Dataset of RDF Licenses," in *Legal Knowledge and Information Systems Conference (ICLKIS)*, 2014.
11. "Creative Commons licenses in RDF." `https://github.com/creativecommons/cc.licenserdf`. Accessed: 2018-11-26.
12. G. Havur, S. Steyskal, O. Panasiuk, A. Fensel, V. Mireles, T. Pellegrini, T. Thurner, A. Polleres, and S. Kirrane, "DALICC: A Framework for Publishing and Consuming Data Assets Legally," in *International Conference on Semantic Systems (SEMANTICS),Poster&Demo*, 2018.
13. M. Mesiti, P. Perlasca, and S. Valtolina, "On the Composition of Digital Licenses in Collaborative Environments," in *Conference on Database and Expert Systems Applications (DEXA)*, 2013.
14. V. Soto-Mendoza, P. Serrano-Alvarado, E. Desmontils, and J. A. Garcia-Macias, "Policies Composition Based on Data Usage Context," in *Workshop Consuming Linked Data (COLD) collocated with ISWC*, 2015.
15. E. Daga, M. d'Aquin, E. Motta, and A. Gangemi, "A Bottom-up Approach for Licences Classification and Selection," in *Workshop on Legal Domain and Semantic Web Applications collocated with ESWC*, 2015.
16. B. A. Davey and H. A. Priestley, *Introduction to Lattices and Order*. Cambridge university press, 2002.